# Diversity is All You Need: Learning Skills without a Reward Function

**Benjamin Eysenbach** [1 *] **Abhishek Gupta** [2] **Julian Ibarz** [1] **Sergey Levine** [1 2]

## Abstract

Intelligent creatures can explore their environments and learn useful skills without supervision. In this paper, we propose DIAYN ("Diversity is All You Need"), a method for learning useful skills without a reward function. Our proposed method learns skills by maximizing an information theoretic objective using a maximum entropy policy. On a variety of simulated robotic tasks, we show that this simple objective results in the unsupervised emergence of diverse skills, such as walking and jumping. In a number of reinforcement learning benchmark environments, our method is able to learn a skill that solves the benchmark task despite never receiving the true task reward. In these environments, some of the learned skills correspond to solving the task, and each skill that solves the task does so in a distinct manner. Our results suggest that unsupervised discovery of skills can serve as an effective pre-training mechanism for overcoming challenges of exploration and data efficiency in reinforcement learning.

## 1. Introduction

Deep reinforcement learning (RL) has been demonstrated to effectively learn a wide range of reward-driven skills, including playing games (Mnih et al., 2013; Silver et al., 2016), controlling robots (Gu et al., 2017; Schulman et al., 2015), and navigating complex environments (Zhu et al., 2017; Mirowski et al., 2016). However, intelligent creatures can explore their environments and learn useful skills even without supervision, so that when they are later faced with specific goals, they can use those skills to satisfy the new goals quickly and efficiently.

Learning skills without reward has several practical applications beyond creating fun videos[1]. Environments with sparse rewards effectively have no reward until the agent randomly reaches a goal state. Learning useful skills without supervision may help address challenges in exploration in these environments. For long horizon tasks, skills discovered without reward could serve as primitives for hierarchical RL, effectively shortening the episode length. In many practical settings, interacting with the environment is essentially free, but evaluating the reward requires human feedback (Christiano et al., 2017). Unsupervised learning of skills may reduce the amount of supervision necessary to learn a task. While we can attempt to take the human out of the loop by designing a reward function, it is challenging to design a reward function that elicits the desired behaviors from the agent (Hadfield-Menell et al., 2017). Finally, when given an unfamiliar environment, it is challenging to determine the range of tasks an agent should be able to learn. Unsupervised skill discovery partially answers this question.

Autonomous acquisition of useful skills without any reward signal is an exceedingly challenging problem. A *skill* is simply a policy. We consider the setting where the reward function is unknown, so we want to learn a set of skills by maximizing the utility of this set. Making progress on this problem requires specifying a learning objective that ensures that each skill individually is distinct and that the skills collectly explore large parts of the state space. In this paper, we show how a simple objective based on mutual information can enable reinforcement learning agents to autonomously discover such skills. We also show different ways in which the skills can be used after being learned.

We propose a method for learning diverse skills with deep RL in the absence of any rewards. We hypothesize that in order to acquire skills that are useful, we must train the skills so that they maximize coverage over the set of possible behaviors. While one skill might perform a useless behavior like random dithering, other skills should perform behaviors that are distinguishable from random dithering, and therefore more useful. A key idea in our work is to use discriminability between skills as an objective. The mutual information between skills and the states they visit encodes

[*]Work done as a member of the Google Brain Residency Program (g.co/brainresidency)
[1]Google Brain, Mountain View, CA, USA
[2]UC Berkeley, Berkeley, CA, USA
Correspondence: Benjamin Eysenbach (eysenbach@google.com)

[1]Videos and code for our experiments will be available at: https://sites.google.com/view/diayn

this idea, and suggests that we should be able to control the states an agent visits by specifying a skill. However, the range of states we can induce the agent to visit is limited by the diversity of our skills. Further, skills that are distinguishable are not necessarily maximally diverse – a slight difference in states makes two skills distinguishable, but not necessarily diverse in a semantically meaningful way. To combat problem, we want to learn skills that not only are distinguishable, but also are *as diverse as possible.* By learning distinguishable skills that are as random as possible, we can "push" the skills away from each other, not only making each skill robust to perturbations, but also more effectively exploring the environment. By maximizing this objective, we can learn skills that run forward, do backflips, skip backwards, and perform face flops (see, e.g., Figure 2).

Our paper makes four contributions. First, we propose a method for learning useful skills without any rewards. We formalize our discriminability goal as maximizing an information theoretic objective with a maximum entropy policy. Second, we show that this simple exploration objective results in the unsupervised emergence of diverse skills, such as running and jumping, on several simulated robotic tasks. In a number of reinforcement learning benchmark environments, our method is able to solve the benchmark task despite never receiving the true task reward. In these environments, some of the learned skills correspond to solving the task, and each skill that solves the task does so in a distinct manner. Third, we show how skills discovered without reward can quickly be adapted to solve a new task. Finally, we show how skills discovered can be used for imitation learning.

## 2. Related Work

Previous work on hierarchical RL has learned skills to maximize a single, known, reward function. Many works (e.g., Bacon et al. (2017); Heess et al. (2016); Dayan & Hinton (1993); Frans et al. (2017); Krishnan et al. (2017); Florensa et al. (2017)) jointly learn a set of skills and a meta-policy for controlling these skills, with the aim of solving a specific task. One problem with joint training (also noted by Shazeer et al. (2017)) is that the meta-policy does not select "bad" options, so these options do not receive any reward signal to improve. Our work prevents this degeneracy by using a random meta-policy during unsupervised skill-learning, such that neither the skills nor the meta-policy are aiming to solve any single task. In contrast to these works, our approach learns skills *with no reward*. Eschewing a reward function not only avoids the difficult problem of reward design, but also allows our method to learn more diverse skills. Our approach can learn skills that are useful for multiple tasks, not just the task specified by the reward function.

Related work has also examined connections between reinforcement learning and information theory. Ziebart et al. (2008) use a probabilistic framework to motivate maximum entropy reinforcement learning. More recently, Schulman et al. (2017) and Haarnoja et al. (2017) show that soft Q learning is equivalent to policy gradient, while Nachum et al. (2017) draw connections between maximum entropy and temporal consistency. Haarnoja et al. (2017) and Haarnoja et al. (2018) propose algorithms for maximum entropy reinforcement learning. We use soft actor critic (Haarnoja et al., 2018) in our experiments.

Recent work has also applied tools from information theory to skill discovery. Mohamed & Rezende (2015) and Jung et al. (2011) use the mutual information between states and actions as a notion of empowerment for an intrinsically motivated agent. Our method maximizes the mutual information between states and *skills*, which can be interpreted as maximizing the empowerment of a *hierarchical agent* whose action space is the set of skills. Hausman et al. (2018), Florensa et al. (2017), and Gregor et al. (2016), and showed that a discriminability objective is equivalent to maximizing the mutual information between the latent skill $z$ and some aspect of the corresponding trajectory. Hausman et al. (2018) considered the setting with many tasks and reward functions and Florensa et al. (2017) considered the setting with a single task reward. Similar to our work, variational intrinsic control (Gregor et al., 2016) learns skills in the *reward-free* setting by maximizing a mutual information term. Three important distinctions allow us to apply our method to tasks significantly more complex than the gridworlds in Gregor et al. (2016). First, we use maximum entropy policies to force our skills to be diverse. Our theoretical analysis shows that including entropy maximization in the RL objective results in the mixture of skills being maximum entropy in aggregate. Second, we fix the distribution $p(z)$ rather than learning it, preventing $p(z)$ from collapsing to sampling only a handful of skills. Third, while the discriminator in Gregor et al. (2016) only looks at the final state, our discriminator looks at every state, which provides additional reward signal. These three crucial differences help explain how our method learns useful skills in complex environments, overcoming limitations in previous work.

Work concurrent with ours draws ties between learning discriminable skills and variational autoencoders. Achiam et al. (2017) use an approach similar to ours to learn skills without reward. Algorithmic design choices, such as our use of an off-policy RL algorithm and conditioning the generator on individual states (rather than full trajectories) allow our method to scale to more complex tasks.

## 3. Diversity is All You Need

We consider an unsupervised RL paradigm in this work, where the agent is allowed an unsupervised "exploration"

**Algorithm 1** DIAYN

**Input:** skill distribution $p(z)$
**repeat**
  Sample skill $z \sim p(z)$ and initial state $s_0 \sim p_0(s)$.
  **for** $t = 1$ **to** *steps_per_episode* **do**
    Sample action $a_t \sim \pi_\theta(a_t \mid s_t, z)$ from skill.
    Step environment: $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$.
    Compute $q_\phi(z \mid s_{t+1})$ with discriminator.
    Set skill reward $r_t = \log q_\phi(z \mid s_{t+1}) - \log p(z)$
    Update policy ($\theta$) to maximize $r_t$ with SAC.
    Update discriminator ($\phi$) with SGD.
  **end for**
**until**

stage followed by a supervised stage. In the unsupervised "exploration" stage, the agent explores the environment, but does not receive any task reward. During the supervised stage, the agent does receive the task reward, and its goal is to learn the task by maximizing the task reward. In our work, the aim of the unsupervised stage is to learn skills that eventually will make it easier to maximize the task reward in the supervised stage. Conveniently, because skills are learned without a priori knowledge of the task, the learned skills can be used for many different tasks. Even if the reward function is known, the unsupervised stage may still be helpful for learning useful skills. Skills learned during unsupervised pretraining may enable an agent to maximize the task reward with fewer samples.

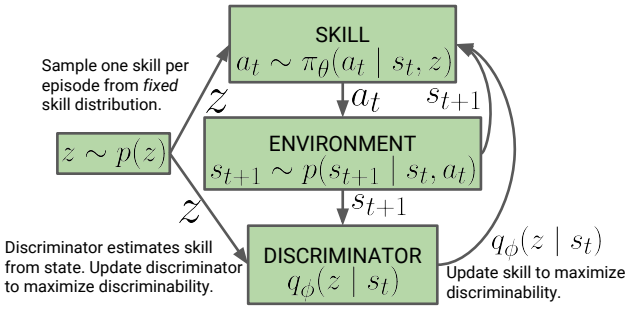



*Figure 1.* **DIAYN**: We sample a skill $z$ at the start of each episode. The discriminator uses states visited to infer which skill generated the trajectory. We update the discriminator to better predict the skill, and update the skill to visit diverse states that make it more discriminable.

### 3.1. How it Works

Our method for unsupervised skill discovery, DIAYN ("Diversity is All You Need"), learns useful skills that are both discriminable and have high entropy. DIAYN builds off of three ideas. First, for skills to be useful, we want the skill to dictate the states that the agent visits. Different skills should visit different states, and hence be distinguishable. Second, we want to use states, not actions, to distinguish skills, because actions that do not affect the environment are not visible to an outside observer. For example, an outside observer cannot tell how much force a robotic arm applies when grasping a cup if the cup does not move. Finally, we encourage exploration and incentivize the skills to be as diverse as possible by learning skills that act as randomly as possible. Skills with high entropy that remain discriminable must explore a part of the state space far away from other skills, lest the randomness in its actions lead it to states where it cannot be distinguished from another skill.

We construct our objective as follows. We maximize the mutual information between skills and states, $MI(s, z)$, to encode the idea that the skill should control which states the agent visits. Conveniently, this mutual information term also encodes the idea that we should be able to infer the skill from the states visited. To ensure that states, not actions, are used to distinguish skills, we minimize the mutual information between skills and actions given the state, $MI(a, z \mid s)$. We can view the collection of skills together with $p(z)$ as a mixture of policies, and we want this mixture policy to also be a maximum entropy policy. Thus, we want to maximize $\mathcal{H}[a \mid s]$. In summary, we want to maximize the following objective:

$$\mathcal{F}(\theta) \triangleq MI(s, z) + \mathcal{H}[a \mid s] - MI(a, z \mid s) \tag{1}$$

Rewriting this objective (proof in Appendix A) gives intuition on how we might optimize it:

$$\mathcal{F}(\theta) = \mathcal{H}[a \mid s, z] + \mathcal{H}[z] - \mathcal{H}[z \mid s] \tag{2}$$

The first term suggests that each skill should act as randomly as possible. Our approach achieves this by using a maximum entropy policy to represent each skill. The second term encourages our prior distribution over $p(z)$ to have high entropy. We fix $p(z)$ to be uniform in our approach, guaranteeing that is has maximum entropy. The third term suggests that it should be easy to infer the skill $z$ from the current state. We cannot compute this directly because it requires integrating over all states and skills. Instead, we approximate the posterior $p(z \mid s)$ with a learned discriminator $q_\phi(z \mid s)$. Jensen's Inequality tells us that replacing $p(z \mid s)$ with $q_\phi(z \mid s)$ gives us a variational lower bound $\mathcal{G}(\theta, \phi)$ on our objective $\mathcal{F}(\theta)$:

$$\begin{aligned}
\mathcal{F}(\theta) &= \mathcal{H}[a \mid s, z] - \mathcal{H}[z \mid s] + \mathcal{H}[z] \\
&= \mathcal{H}[a \mid s, a] + \mathbb{E}[\log p(z \mid s)] - \mathbb{E}[\log p(z)] \\
&\geq \mathcal{H}[a \mid s, z] + \mathbb{E}[\log q_\phi(z \mid s) - p(z)] \triangleq \mathcal{G}(\theta, \phi)
\end{aligned}$$

The expectation is taken with a skill $z$ sampled from the prior distribution over skills $p(z)$ and with states $s$ sampled from the distribution over states $p(s \mid z)$ induced by skill $z$.

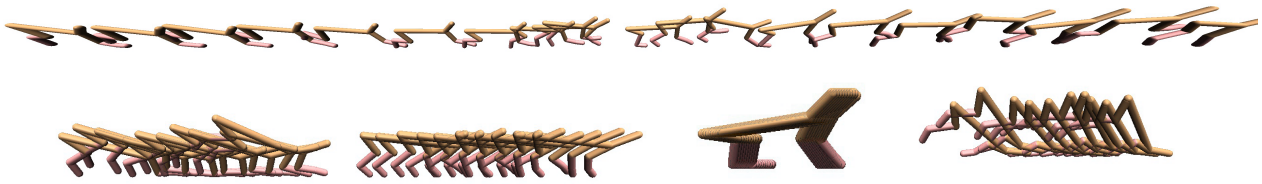


*Figure 2.* **Half cheetah skills** Without any reward, half cheetah learns (top row) to run backwards, to run forward, (bottom row) to inch backwards, to tiptoe forward, to glide forward, and to scoot forward on its nose. In our experiments, the majority of the skills involved continual motion rather than assuming static poses. To the best of our knowledge, it is not commonly known that half cheetah could acquire the skills in the bottom row. This observation motivates our claim that human designers simply do not know or cannot imagine some skills agents can acquire.

### 3.2. Implementation

We implement DIAYN with soft actor critic (Haarnoja et al., 2018). We use soft actor critic to learn a policy $\pi_\theta(a \mid s, z)$ that is conditioned on the skill $z$. Note that we use "skill" to refer to both the latent variable $z$ and the policy conditioned on $z$. Soft actor critic maximizes the policy's entropy over actions, which takes care of the entropy term in our objective $\mathcal{G}$. Following Haarnoja et al. (2018), we scale the entropy regularizer $\mathcal{H}[a \mid s, a]$ by $\alpha$. We found empirically that an $\alpha = 0.1$ provided a good trade-off between exploration and discriminability. We maximize the expectation in $\mathcal{G}$ by replacing the task reward with the following pseudo-reward:

$$r_z(s, a) \triangleq \log q_\phi(z \mid s) - \log p(z) \tag{3}$$

We discuss the importance of the $\log p(z)$ term in Appendix B.

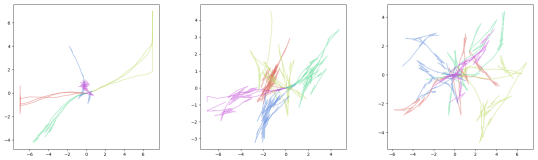
Algorithm 1 provides pseudo code for DIAYN. During unsupervised learning, we sample a skill $z \sim p(z)$ at the start of each episode, and act according to that skill throughout the episode. The agent is rewarded for visiting states that are easy to discriminate, while the discriminator is updated to better infer the skill $z$ from states visited. Entropy regularization occurs as part of the SAC update.

## 4. What Skills are Learned?

In this section, we examine the learned skills to answer three questions about the skills learned by DIAYN. We highly encourage readers to view videos of the learned skills at the following website: `https://sites.google.com/view/diayn`

**Question 1** *Does entropy regularization lead to more diverse skills?*

To answer this question, we apply our method to a 2D point mass. The agent controls the orientation and forward velocity of the point, with is confined within a 2D box. We vary the entropy regularization $\alpha$, with larger values of $\alpha$ corresponding to policies with more stochastic actions. Figure 3 shows skills learned for three values of $\alpha$. With small values of $\alpha$ (left), we learn skills that move large distances in different directions but fail to explore large parts of the state space. Increasing $\alpha$ (middle) makes the skills visit a more diverse set of states, which may help with exploration in complex state spaces. Increasing $\alpha$ too much (right) makes it difficult to discriminate the skills.



$\alpha = 0.01$ $\alpha = 1$ $\alpha = 10$

*Figure 3.* Illustration of 5 skills learned in a simple 2D navigation environment. Increasing the entropy regularizer $\alpha$ makes each skill visit a larger set of states, yet can make discriminating the skills more difficult.

*Figure 4.* **Hopper skills**: Without any reward, hopper learns (top row) to hop forward, to hop backwards, (bottom row) to shuffle backwards, to hop in place, to lean forwards, and to dive forwards. The pseudo-reward in Equation 3 is positive, incentivizing to avoid falling, which terminates the episode in the standard hopper benchmark task.

**Question 2** *Does our method learn diverse skills for simulated robots?*

We apply DIAYN to three standard RL benchmark environments: half-cheetah, hopper, and ant. In all environments, we learn diverse locomotion primitives, as shown in Figures 2, 4, 5. Despite never receiving any reward, the half cheetah and hopper learn skills that move forward and

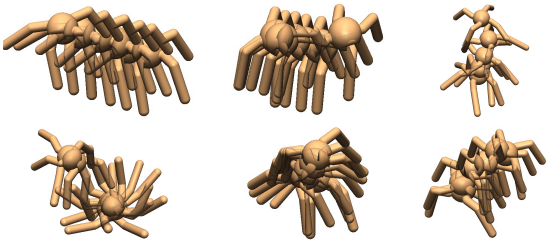


*Figure 5.* **Ant skills**: Ant learns (clockwise from top left) to walk left, to walk right, to walk down, to walk up, to turn in place, and to flip on its back.

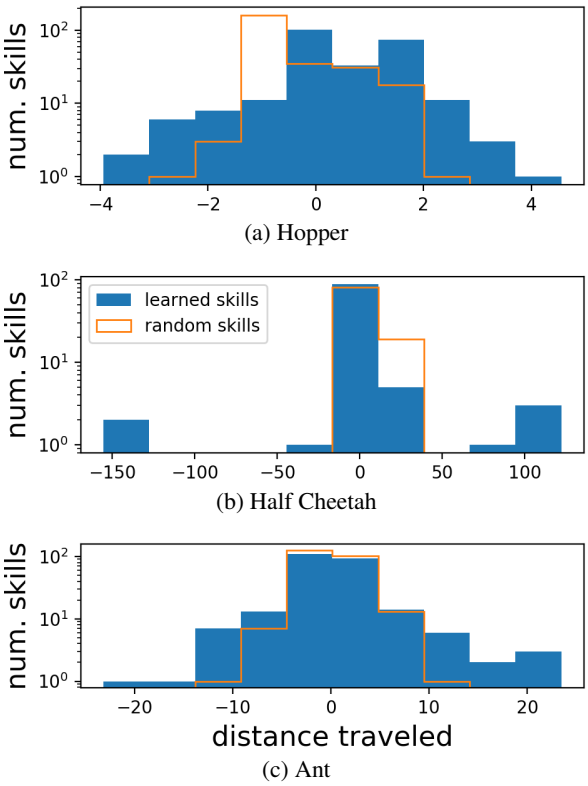
achieve large task reward on the corresponding RL benchmarks, which all require them to move forward at a fast pace. Perhaps unsurprisingly, half cheetah and hopper also learn skills that move backwards, corresponding to receiving a task reward much smaller than what a random policy would receive. Unlike hopper and half cheetah, the ant is free to move in the XY plane. While it learns skills that move in different directions, most skills move in arcs rather than straight lines, meaning that we rarely learn a single skill that achieves large task reward on the typical task of running forward. In the appendix, we visualize the objective throughout training.

Figure 6 shows the distance traveled by each skill along the X-axis. The wide distribution over distance traveled demonstrates the diversity of our learned skills. Additionally, in all environments, some skills correspond to moving forward, while others move backwards, demonstrating that our skills are not aimed at solving any particular task. Given a task description (e.g., run forward), we can easily find a skill that performs this task. We visualize and discuss the distribution over task rewards in Appendix F.

**Question 3** *Does our method learn useful skills?*

Experiments on half cheetah and hopper (discussed above) show that, among the skills learned by our method, there is often a skill that runs forward quickly and achieves a high reward on the standard benchmark reward function for the environment, despite never having seen that reward function during training. We also apply DIAYN to two classic control tasks, inverted pendulum and mountain car. In the mountain car task, the agent controls the X coordinate of a car stuck between two valleys. The goal is to escape up and over the mountain on the right side of the valley. The car does not have enough power to directly drive up the mountain, so it much first gather momentum by rocking back and forth across the valley. In the inverted pendulum task, the agent balances an inverted pendulum by controlling its base.

On both tasks, among the skills learned by our method are skills that actually solve the standard benchmark task, despite never having seen the standard benchmark reward




(a) Hopper

(b) Half Cheetah

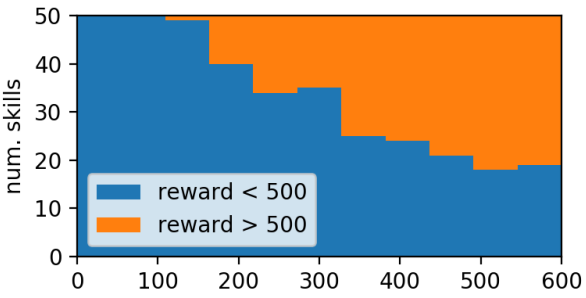
(c) Ant

*Figure 6.* **Distance traveled by learned skills**: We show the distance traveled by our learned skills. Note that on all tasks, the learned skills move further (either forward or backwards) than random skills.

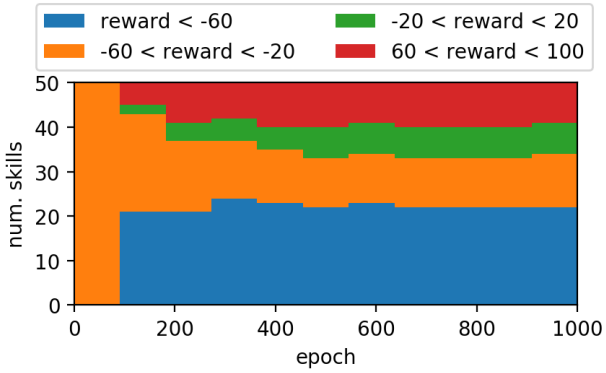
function during training. While the skills are learned without any reward, we can evaluate each skill with reward. Figure 7 shows how the learned skills become increasingly diverse throughout training. On the inverted pendulum (Fig. 7a), initially no skills balance the pendulum upright. As the skills are optimized to become as diverse as possible, this distribution shifts towards having half the skills balance the pendulum. We observe a similar pattern for mountain car (Fig. 7b). Recall that our skills are learned with no reward, so it is natural that some skills correspond to small task reward while others correspond to large task reward. Not only does our approach learn skills that solve the task without rewards, it learns multiple distinct skills for solving the task. For inverted pendulum, different skills balance the pendulum at different positions with different frequencies. For mountain car, skills that climb the mountain employ a variety of strategies for to do so. We visualize these skills in Appendix G.1.

## 5. Harnessing Learned Skills

Now that we have a set of skills, how can we tell the agent to do something useful with them? The simplest approach,

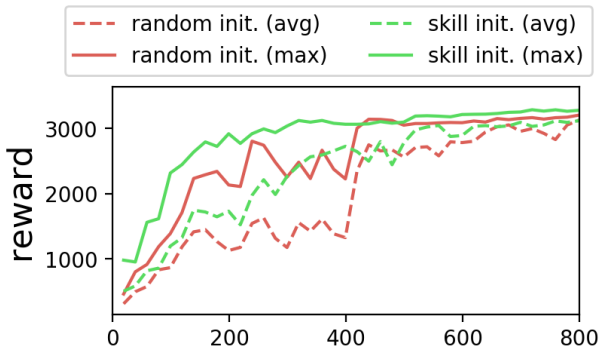



(a) Inverted Pendulum

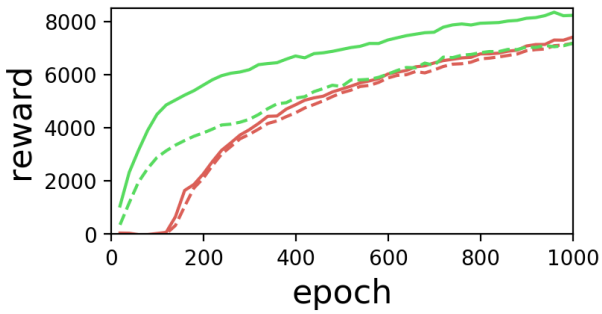



(b) Mountain Car

*Figure 7.* **Diversity of Rewards**: While are skills are learned without reward, evaluating their task reward gives an indicator of their diversity. On both inverted pendulum (top) and mountain car (bottom), the diversity of the 50 skills increases throughout training.




(a) Hopper




(b) Half cheetah

*Figure 8.* **Maximizing a reward**: We maximize the task reward on (top) hopper, and (bottom) half cheetah. For random initialization, we plot the average and best seed, while for skill initialization, we plot the average and best skill initialization from multiple unsupervised training runs. Note that finetuning skills learned with unsupervised skill learning achieves substantially faster learning than random initialization.

as discussed in the previous section, is to simply pick the best skill for a given reward. In this section, we present two methods for using the learned skills for solving new tasks, and design experiments to answer three questions about DIAYN. As discussed in Section 1, there are many other applications for skills we learn through unsupervised exploration.

### 5.1. Maximizing a Reward

First, we consider the setting where we first learn skills without supervision from the task reward, and then quickly adapt our skills to perform a desired task by maximizing the task reward. In this section, we show that this kind of "unsupervised pretraining" allows us to learn much more quickly once the reward function is provided. This motivates using our algorithm in resource-constrained settings where the reward function is expensive to compute. In environments with many potential tasks and reward functions, DIAYN provides a task-independent initialization for solving any of the tasks. Our approach is similar to how many computer vision models are initialized from a model trained on ImageNet, with the added benefit that we do not need a large, labeled training set.

**Question 4** *Can we use learned skills to directly maximize the task reward?*

In this section, we use the skills learned during pretraining to efficiently maximize the task reward, as specified by a reward function. After the unsupervised pretraining step, we choose the skill that has the highest reward. We then further finetune this skill using the task-specific reward function. We compare to a "random initialization" baseline that is initialized from scratch. Our "skill initialization" approach differs from this baseline only in how weights are initialized – for both, we use soft actor critic with the same hyperparameters. For the "skill initialization" approach, we initialize both the actor and critic networks with weights learned during unsupervised pretraining. Although the critic networks learned during pretraining corresponds to the pseudo-reward from the discriminator (Eq. 3) and not the true task reward, we found empirically that the discriminator pseudo-reward was close to the true task reward for the best skill, and initialization the critic in addition to the actor further sped up learning.

Figure 8 shows both methods applied to half cheetah and hopper. We plot the average over all sets of skills, as well as finetuning of the best unsupervised skill (obtained from N separate unsupervised training runs). Note that finetuning the best skill is *not* equivalent to picking the best finetuning random seed – if we assume that unsupervised training is cheap compared to training with rewards, we can easily run the unsupervised training as many times as desired to obtain the best initialization. On both tasks, we observe that our unsupervised pretraining enables the agent to learn the task reward more quickly. As expected, finetuning does not change the final task reward achieved at the end of training. Thus, finetuning a learned skill allows an agent to learn a new task quickly, with fewer iterations and less data.

### 5.2. Imitating an Expert

Demonstration is often a natural way to specify a task to an agent. One use-case is where a human manually controls the agent to complete a task that we would like to automate. Simply replaying the human's actions fails in stochastic environments, cases where closed-loop control is necessary. A second use-case involves an existing agent with a hard coded, manually designed policy. Imitation learning replaces the existing policy with a similar yet differentiable policy, which might be easier to update in response to new constraints or objectives.

We consider the setting where we are given an expert trajectory consisting of states (not actions): $\tau^* = \{(s_i)\}_{1 \le i \le N}$. We want to obtain a feedback controller that will reach the same states. We propose a simple method for using skills learned during unsupervised pretraining to imitate an expert trajectory $\tau^*$. Given the expert trajectory, we use our learned discriminator to estimate which skill was most likely to have generated the trajectory:

$$\hat{z} = \arg\max_z \Pi_{s_t \in \tau^*} q_\phi(z \mid s_t)$$

As motivation for this optimization problem, note that each skill induces a distribution over states, $p^z \triangleq p(s \mid z)$. We use $p^*$ to denote the distribution over states for the expert policy. With a fixed prior distribution $p(z)$ and a perfect discriminator $q_\phi(z \mid s) = p(z \mid s)$, we have $p(s \mid z) \propto q_\phi(z \mid s)$ as a function of $z$. Thus, Equation 5.2 is an M-projection of the expert distribution over states onto the family of distributions over states, $\mathcal{P} = \{p^z\}$:

$$\arg\min_{p^z \in \mathcal{P}} D(p^* \mid\mid p^z) \tag{4}$$

For clarity, we omit a constant that depends only on $p^*$. Note that the use of an M-projection, rather than an I-projection, helps guarantee that the retrieved skill will visit all states that the expert visits (Christopher, 2016). In our experiments, we solve Equation 4 by simply iterating over skills.

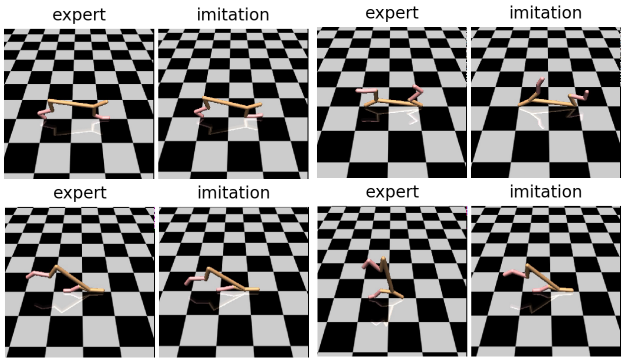



*Figure 9.* **Cheetah see, cheetah do**: Our learned skills can imitate an expert standing upright (top left), flipping onto its back (top right), rolling onto its nose (bottom left). The bottom right figure shows a failure case, were our method fails imitate a handstand.

**Question 5** *Can we use learned skills to imitate an expert?*

We perform two experiments using this imitation method. First, we qualitatively evaluate this approach to imitation learning on half cheetah. Figure 9 shows four imitation tasks, three of which our method successfully imitates. Note that some of these tasks (and those in Figure 2) would be challenging to describe with a reward function. For many, it is easier to obtain a demonstration of the expert trajectory. A human tasked with teaching the desired task to an agent might even look at all learned skills and choose whichever corresponds most closely with the desired task.

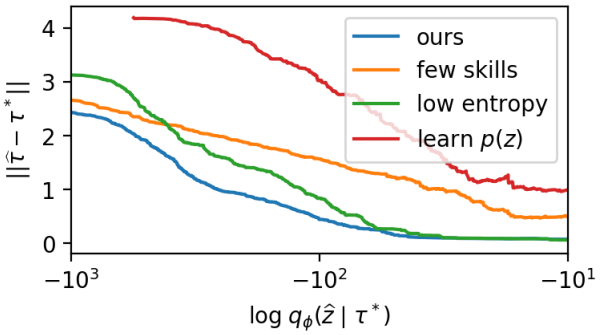



*Figure 10.* **Imitating an expert**: We show aggregate results from many imitation tasks. The X-axis shows the discriminator's prediction of how well the skill we choose will match the expert, while the Y-axis shows the true distance between expert trajectory and imitation trajectory, averaged across all tasks with greater or equal log likelihood. For all tasks, the trajectory distances decreases as the discriminator's prediction increases, indicating that the discriminator is a good predictor of whether we will succeed at the imitation task. Our method achieves a smaller trajectory distance than all baselines.

We next quantitatively evaluate this imitation method on classic control tasks. The "expert" trajectories are actually generated synthetically in these experiments, by running a

different random seed of our algorithm. A different seed is used to ensure that the trajectories are not actually produced by any of the currently available skills. Of course, in practice, the expert trajectories might be provided by any other means, including a human. For each expert trajectory, we retrieve the closest student skill $\hat{z}$ using Equation 5.2. Evaluating $q_\phi(\hat{z} \mid \tau^*)$ gives us an estimate of the probability that the imitation will match the expert. This quantity is useful for predicting how accurately our method will imitate an expert before executing the imitation policy. In a safety critical setting, a user may avoid attempting tasks where this score is low.

We apply this approach to inverted pendulum and continuous mountain car. We compare our method to three baselines. The "low entropy" baseline is a variant on our method with lower entropy regularization. The "learned $p(z)$" baseline learns the distribution over skills. Note that Variational Intrinsic Control (Gregor et al., 2016) is a combination of the "low entropy" baseline and the "learned $p(z)$" baseline. Finally, the "few skills" baseline learns only 5 skills, whereas all other methods learn 50.

Figure 10 shows the results aggregated across 600 imitation tasks. The X-axis shows the discriminator score, our estimate for how well the imitation policy will match the expert. The Y-axis shows the true distance between the trajectories, as measured by L2 distance in state space. For all methods, the distance between the expert and the imitation decreases as the discriminator's score increases, indicating that the discriminator's score is a good predictor of task performance. Second, our method consistently achieves the lowest trajectory distance among all methods. The "low entropy" baseline is slightly worse, motivating our decision to learn maximum entropy skills. When imitating tasks using the "few skills" baseline, the imitation trajectories are even further from the expert trajectory. This is expected – by learning more skills, we obtain a better "coverage" over the space of skills. A "learn $p(z)$" baseline that learns the distribution over skills also performs poorly. Recalling that Gregor et al. (2016) is a combination of the "low entropy" baseline and the "learn $p(z)$" baseline, this plot provides evidence that using maximum entropy policies and fixing the distribution for $p(z)$ are two factors that enabled our method to scale to more complex tasks.

**Question 6** *What is the effect of learning the distribution over skills, $p(z)$?*

While previous work (Gregor et al., 2016) has learned the distribution over skills, we fix the distribution over skills. We found empirically that leaving this distribution fixed allows us to discover more diverse skills. We include a "learn $p(z)$" baseline in the imitation experiment in Figure 10. This baseline was significantly worse at imitating an expert trajectory, suggesting that learning $p(z)$ results in learning fewer diverse skills. If few diverse skills are learned, it is challenging to imitate an arbitrary expert policy. To further analyze this, we visualize the entropy of $p(z)$ when learning skills for half cheetah and the classic control tasks. Figure 11 shows that learning $p(z)$ decreases the effective number of skills sampled by 10x. Appendix E.2 explains how we compute the effective number of skills, and shows that the classic control tasks encounter the same limitation.

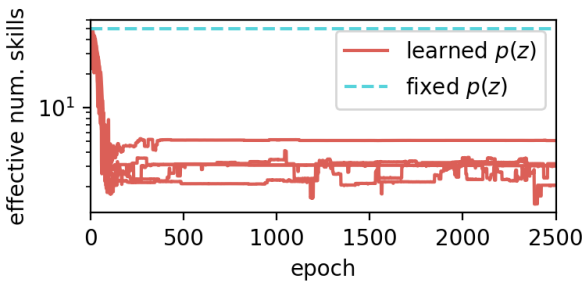



*Figure 11.* **Effect of learning $p(z)$**: We plot the effective number of skills that are sampled from the skill distribution $p(z)$ throughout training. Note how learning $p(z)$ greatly reduces the effective number of skills. We show results from 5 random seeds for each experiment on the half cheetah environment.

## 6. Conclusion

In this paper, we presented DIAYN, a method for learning skills without reward functions. We showed that DIAYN learns diverse skills for complex tasks, often solving benchmark tasks with one of the learned skills without actually receiving any task reward. We further proposed methods for leveraging the learned skills to maximize a reward function and to imitate an expert. Looking ahead, DIAYN may make learning a task easier by replacing the task's complex action space with a set of useful skills. DIAYN could be combined with methods for augmenting the observation space and reward function. Using the common language of information theory, a theoretically grounded, joint objective can likely be derived. Another research direction is using unsupervised skill discovery to solicit human preferences. Building upon Christiano et al. (2017), we can likely infer human preferences more quickly by having humans select among learned skills. Finally, from a perspective of creativity and education, the skills produced by DIAYN might be used by game designers to allow players to control complex robots and by artists to design dancing robots. Both these efforts might introduce the challenges and opportunities of RL to an audience outside the ML community.

## References


Achiam, Joshua, Edwards, Harrison, Amodei, Dario, and Abbeel, Pieter. Variational autoencoding learning of options by reinforcement. *NIPS Deep Reinforcement Learning Symposium*, 2017.

Bacon, Pierre-Luc, Harb, Jean, and Precup, Doina. The option-critic architecture. In *AAAI*, pp. 1726–1734, 2017.

Christiano, Paul F, Leike, Jan, Brown, Tom, Martic, Miljan, Legg, Shane, and Amodei, Dario. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pp. 4302–4310, 2017.

Christopher, M Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2016.

Dayan, Peter and Hinton, Geoffrey E. Feudal reinforcement learning. In *Advances in neural information processing systems*, pp. 271–278, 1993.

Duan, Yan, Chen, Xi, Houthooft, Rein, Schulman, John, and Abbeel, Pieter. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pp. 1329–1338, 2016.

Florensa, Carlos, Duan, Yan, and Abbeel, Pieter. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

Frans, Kevin, Ho, Jonathan, Chen, Xi, Abbeel, Pieter, and Schulman, John. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*, 2017.

Gregor, Karol, Rezende, Danilo Jimenez, and Wierstra, Daan. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Gu, Shixiang, Holly, Ethan, Lillicrap, Timothy, and Levine, Sergey. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 3389–3396. IEEE, 2017.

Haarnoja, Tuomas, Tang, Haoran, Abbeel, Pieter, and Levine, Sergey. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

Haarnoja, Tuomas, Zhou, Aurick, Abbeel, Pieter, and Levine, Sergey. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Hadfield-Menell, Dylan, Milli, Smitha, Abbeel, Pieter, Russell, Stuart J, and Dragan, Anca. Inverse reward design. In *Advances in Neural Information Processing Systems*, pp. 6768–6777, 2017.

Hausman, Karol, Springenberg, Jost Tobias, Wang, Ziyu, Heess, Nicolas, and Riedmiller, Martin. Learning an embedding space for transferable robot skills. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rk07ZXZRb.

Heess, Nicolas, Wayne, Greg, Tassa, Yuval, Lillicrap, Timothy, Riedmiller, Martin, and Silver, David. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

Henderson, Peter, Islam, Riashat, Bachman, Philip, Pineau, Joelle, Precup, Doina, and Meger, David. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

Jung, Tobias, Polani, Daniel, and Stone, Peter. Empowerment for continuous agentenvironment systems. *Adaptive Behavior*, 19 (1):16–39, 2011.

Krishnan, Sanjay, Fox, Roy, Stoica, Ion, and Goldberg, Ken. Ddco: Discovery of deep continuous options for robot learning from demonstrations. In *Conference on Robot Learning*, pp. 418–437, 2017.

Mirowski, Piotr, Pascanu, Razvan, Viola, Fabio, Soyer, Hubert, Ballard, Andrew J, Banino, Andrea, Denil, Misha, Goroshin, Ross, Sifre, Laurent, Kavukcuoglu, Koray, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, and Riedmiller, Martin. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mohamed, Shakir and Rezende, Danilo Jimenez. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 2125–2133, 2015.

Nachum, Ofir, Norouzi, Mohammad, Xu, Kelvin, and Schuurmans, Dale. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2772–2782, 2017.

Schulman, John, Moritz, Philipp, Levine, Sergey, Jordan, Michael, and Abbeel, Pieter. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Schulman, John, Abbeel, Pieter, and Chen, Xi. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017.

Shazeer, Noam, Mirhoseini, Azalia, Maziarz, Krzysztof, Davis, Andy, Le, Quoc, Hinton, Geoffrey, and Dean, Jeff. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Silver, David, Huang, Aja, Maddison, Chris J, Guez, Arthur, Sifre, Laurent, Van Den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Zhu, Yuke, Mottaghi, Roozbeh, Kolve, Eric, Lim, Joseph J, Gupta, Abhinav, Fei-Fei, Li, and Farhadi, Ali. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 3357–3364. IEEE, 2017.

Ziebart, Brian D, Maas, Andrew L, Bagnell, J Andrew, and Dey, Anind K. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

## A. Proof of Objective Equality

Below, we provide a proof of the equality in Equation 2.

$$\begin{aligned}\mathcal{F}(\theta) &= MI(s,z) + \mathcal{H}[a \mid s] - MI(a,z \mid s) \\ &= (\mathcal{H}[z] - \mathcal{H}[z \mid s]) + \mathcal{H}[a \mid s] \\ &\quad - (\mathcal{H}[a \mid s] - \mathcal{H}[a \mid s,z]) \\ &= \mathcal{H}[z] - \mathcal{H}[z \mid s] + \mathcal{H}[a \mid s,z]\end{aligned}$$

While our method uses stochastic policies, note that for deterministic policies in continuous action spaces, $MI(a,z \mid s) = \mathcal{H}[a \mid s]$. Thus, for deterministic policies, Equation 2 reduces to maximizing $MI(s,z)$.

## B. Pseudo-Reward

Note that $\log p(z)$ in Equation 3 is a baseline that does not depend on the policy parameters $\theta$, so one might be tempted to remove it from the objective. We provide a two justifications for keeping it. First, assume that episodes never terminate, but all skills eventually converge to some absorbing state (e.g., with all sensors broken). At this state, the discriminator cannot distinguish the skills, so its estimate is $\log q(z \mid s) = \log(1/N)$, where $N$ is the number of skills. For practical reasons, we want to restart the episode after the agent reaches the absorbing state. Subtracting $\log(z)$ from the pseudo-reward at every time step in our finite length episodes is equivalent to pretending that episodes never terminate and the agent gets reward $\log(z)$ after our "artificial" termination. Second, assuming our discriminator $q_\phi$ is better than chance, we see that $q_\phi(z \mid s) \geq p(z)$. Thus, subtracting the $\log p(z)$ baseline ensures our reward function is always non-negative, encouraging the agent to stay alive. Without this baseline, an optimal agent would end the episode as soon as possible.[2]

## C. Experimental Details

In our experiments, we use the same hyperparameters as those in Haarnoja et al. (2018), with one notable exception. For the Q function, value function, and policy, we use neural networks with 300 hidden units instead of 128 units. We found that increasing the model capacity was necessary to learn many diverse skills. When comparing the "skill initialization" to the "random initialization" in Section 5, we use the same model architecture for both methods. To pass skill $z$ to the Q function, value function, and policy, we simply concatenate $z$ to the current state $s_t$. As in Haarnoja et al. (2018), epochs are 1000 episodes long. For all environments, episodes are at most 1000 steps long, but may be shorter. For example, the standard benchmark hopper environment terminates the episode once it falls over. Figures 7 and 8 show up to 1000 epochs, which corresponds to at most 1 million steps. We found that learning was most stable when we scaled the maximum entropy objective ($\mathcal{H}[a \mid s,a]$ in Eq. 1) by $\alpha = 0.1$. We use this scaling for all experiments.

[2] In some environments, such as mountain car, it is desirable for the agent to end the episode as quickly as possible. For these types of environments, the $\log p(z)$ baseline can be removed.

## D. Training Objectives

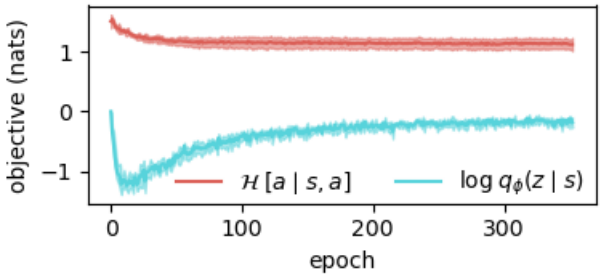



*Figure 12.* **Objectives**: We plot the two terms from our objective (Eq. 1) throughout training. While the entropy regularizer (red) quickly plateaus, the discriminability term (blue) term continues to increase, indicating that our skills become increasingly diverse without collapsing to deterministic policies. This plot shows the mean and standard deviation across 5 seeds for learning 20 skills in half cheetah environment. Note that $\log_2(1/20) \approx -3$, setting a lower bound for $\log q_\phi(z \mid s)$.

To provide further intuition into our approach, Figure 12 plots the two terms in our objective throughout training. Our skills become increasingly diverse throughout training without converging to deterministic policies.

## E. Learning $p(z)$

We used our method as a starting point when comparing to VIC (Gregor et al., 2016) in Section 5. While $p(z)$ is fixed in our method, we implement VIC by learning $p(z)$. In this section, we describe how we learned $p(z)$, and show the effect of learning $p(z)$ rather than leaving it fixed.

### E.1. How to Learn $p(z)$

We choose $p(z)$ to optimize the following objective, where $p_z(s)$ is the distribution over states induced by skill $s$:

$$\begin{aligned}\mathcal{H}[s,z] &= \mathcal{H}[z] - \mathcal{H}[z \mid s] \\ &= \sum_z -p(z)\log p(z) + \sum_z \mathbb{E}_{s\sim p_z(s)}\left[\log p(z \mid s)\right] \\ &= \sum_z p(z)\left(\mathbb{E}_{s\sim p_z(s)}\left[\log p(z \mid s)\right] - \log p(z)\right)\end{aligned}$$

For clarity, we define $p_z^t(s)$ as the distribution over states induced by skill $z$ at epoch $t$, and define $\ell_t(z)$ as an approximation of $\mathbb{E}[\log p(z \mid s)]$ using the policy and discriminator

from epoch $t$:

$$\ell_t(z) \triangleq \mathbb{E}_{s \sim p_z^t(s)}[\log q_t(z \mid s)]$$

Noting that $p(z)$ is constrained to sum to 1, we can optimize this objective using the method of Lagrange multipliers. The corresponding Lagrangian is

$$\mathcal{L}(p) = \sum_z p(z)\left(\ell_t(z) - \log p(z)\right) + \lambda\left(\sum_z p(z) - 1\right)$$

whose derivative is

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial p(z)} &= \cancel{p(z)}\left(\frac{-1}{\cancel{p(z)}}\right) + \ell_t(z) - \log p(z) + \lambda \\ &= \ell_t(z) - \log p(z) + \lambda - 1\end{aligned}$$

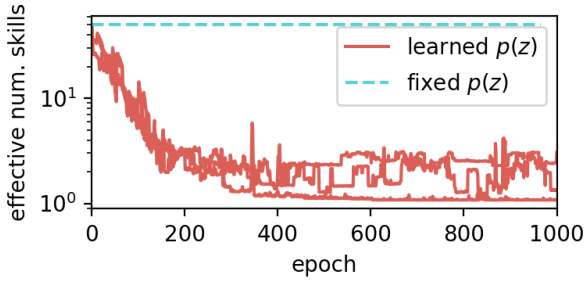
Setting the derivative equal to zero, we get

$$\log p(z) = \ell_t(z) + \lambda - 1$$

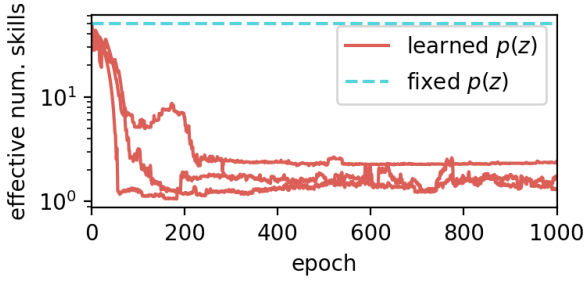
and finally arrive at

$$p(z) \propto e^{\ell_t(z)}$$

### E.2. Effect of Learning $p(z)$




(a) Inverted pendulum




(b) Mountain car

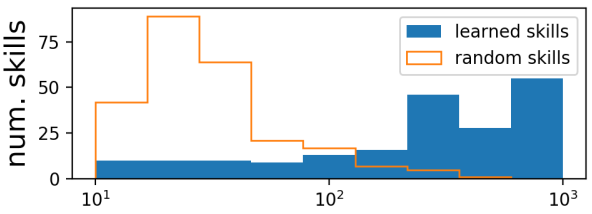
*Figure 13.* **Effect of learning $p(z)$**: We plot the effective number of skills that are sampled from the skill distribution $p(z)$ throughout training. Note how learning $p(z)$ greatly reduces the effective number on inverted pendulum and mountain car. We show results from 3 random seeds for each environment.

In this section, we briefly discuss the effect of learning $p(z)$ rather than leaving it fixed. To study the effect of learning $p(z)$, we compared the entropy of $p(z)$ throughout training. When $p(z)$ is fixed, the entropy is a constant ($\log(50) \approx 3.9$). To convert nats to a more interpretable quantity, we compute the effective number of skills by exponentiation the entropy:

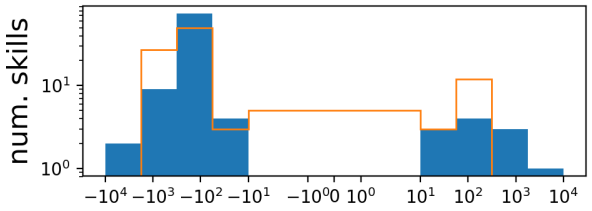
$$\text{effective num. skills} \triangleq e^{\mathcal{H}[z]}$$

Figure 11 plots the effective number of skills throughout training for half cheetah, while Figure 13 shows the results for inverted pendulum, and mountain car. Note how the effective number of skills drops by a factor of 10x when we learn $p(z)$. This observation supports our claim that learning $p(z)$ results in learning fewer diverse skills.

## F. Distribution over Task Reward




(a) Hopper




(b) Half Cheetah


(c) Ant

*Figure 14.* **Task reward of skills learned without reward**: While our skills are learned without the task reward function, we evaluate each with the task reward function for analysis. The wide range of rewards shows the diversity of the learned skills. In the hopper and half cheetah tasks, many skills achieve large task reward, despite not observing the task reward during training. As discussed in prior work (Henderson et al., 2017; Duan et al., 2016), standard model-free algorithms trained directly on the task reward converge to scores of 1000 - 3000 on hopper, 1000 - 5000 on cheetah, and 700 - 2000 on ant.

In Figure 14, we take the skills learned without any rewards, and evaluate each of them on the standard benchmark reward function. We compare to random (untrained) skills. The wide distribution over rewards is evidence that the skills learned are diverse. For hopper, some skills hop or stand for the entire episode, receiving a reward of at least 1000. Other skills aggressively hop forwards or dive backwards, and receive rewards between 100 and 1000. Other skills fall over immediately and receive rewards of less than 100. The benchmark half cheetah reward includes a control penalty for taking actions. Unlike random skills, learned skills rarely have task reward near zero, indicating that all take actions to become distinguishable. Skills that run in place, flop on their nose, or do backflips receive reward of -100. Skills that receive substantially smaller reward correspond to running quickly backwards, while skills that receive substantially larger reward correspond to running forward. Similarly, the benchmark ant task reward includes both a control penalty and a survival bonus, so random skills that do nothing receive a task reward near 1000. While no single learned skill learns to run directly forward and obtain a task reward greater than 1000, our learned skills run in different patterns to become discrminable, resulting in a lower task reward.

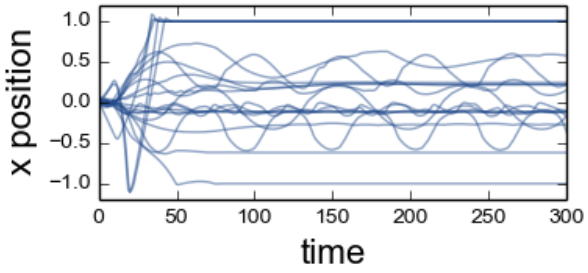
## G. Visualizing Learned Skills

### G.1. Classic Control Tasks

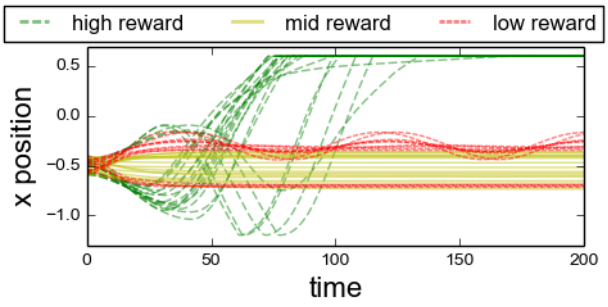
In this section, we visualize the skills learned for inverted pendulum and mountain car without a reward. Not only does our approach learn skills that solve the task without rewards, it learns multiple distinct skills for solving the task. Figure 15 shows the X position of the agent across time, within one episode. For inverted pendulum (Fig. 15a), we plot only skills that solve the task. Horizontal lines with different X coordinates correspond to skills balancing the pendulum at different positions along the track. The periodic lines correspond to skills that oscillate back and forth while balancing the pendulum. Note that skills that oscillate have different X positions, amplitudes, and periods. For mountain car (Fig. 15b), skills that climb the mountain employ a variety of strategies for to do so. Most start moving backwards to gather enough speed to summit the mountain, while others start forwards, then go backwards, and then turn around to summit the mountain. Additionally, note that skills differ in when the turn around and in their velocity (slope of the green lines).

### G.2. Simulated Robot Tasks

Figures 16, 17, and 18 show more skills learned *without reward*.




(a) Inverted Pendulum




(b) Mountain Car

*Figure 15.* **Visualizing Skills**: For every skill, we collect one trajectory and plot the agent's X coordinate across time. For inverted pendulum (top), we only plot skills that balance the pendulum. Note that among balancing skills, there is a wide diversity of balancing positions, control frequencies, and control magnitudes. For mountain car (bottom), we show skills that achieve larger reward (complete the task), skills with near-zero reward, and skills with very negative reward. Note that skills that solve the task (green) employ varying strategies.

*Figure 16.* **Half cheetah skills**: We show skills learned by half-cheetah with no reward.

*Figure 17.* **Hopper Skills**: We show skills learned by hopper with no reward.

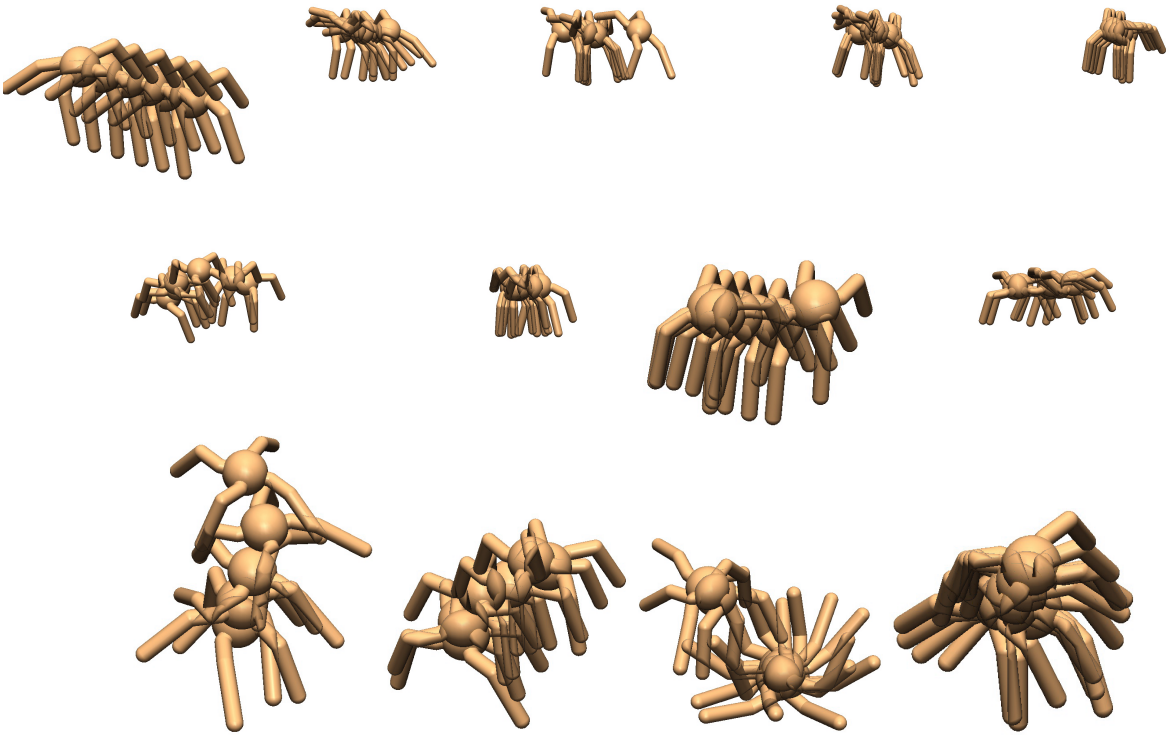


*Figure 18.* **Ant skills**: We show skills the ant learns without any supervision. Ant learns (top row) to move right, (middle row) to move left, (bottom row, left to right) to move up, to move down, to flip on its back, and to rotate in place.